POLITECNICO DI TORINO
Repository ISTITUZIONALE

Enhanced Machine-Learning Flow for Microwave-Sensing Systems for Contaminant Detection in Food

Original
Enhanced Machine-Learning Flow for Microwave-Sensing Systems for Contaminant Detection in Food / Štiti, B., Urbinati, L., Guglielmo, G.D., Carloni, L.P., Casu, M.R.. - In: IEEE TRANSACTIONS ON AGRIFOOD ELECTRONICS.. - ISSN 2771-9529. - ELETTRONICO. - 2:2(2024), pp. 181-189. [10.1109/tafe.2024.3421238]

Availability:
This version is available at: 11583/2994268 since: 2024-11-08T11:29:32Z

Publisher:
IEEE

Published
DOI:10.1109/tafe.2024.3421238

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

17 November 2026

This article has been accepted for publication in IEEE Transactions on AgriFood Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAFE.2024.3421238

Enhanced Machine-Learning Flow for Microwave-Sensing Systems for Contaminant Detection in Food

Bernardita Štitić, Luca Urbinati, *Graduate Student Member, IEEE,* Giuseppe Di Guglielmo, Luca P. Carloni, *Fellow, IEEE,* Mario R. Casu, *Senior Member, IEEE.*

***Abstract*—Combining data-driven Machine Learning (ML) with Microwave Sensing (MWS) makes it possible to analyze packaged food in real-time without any contact and spot low-density contaminants (e.g., plastics or glass splinters) that current industrial food safety methods cannot detect. This is achieved by training ML classifiers on the scattered signal reflected by the target food product exposed to microwaves. In this paper, we present an enhanced ML flow to boost foreign body detection accuracy of ML classifiers in MWS systems. Innovations include assessing the performance of a multi-class classifier, training it with microwave frequency pairs as features, data augmentation, a new pre-processing scaler suitable for the feature distributions in the datasets, quantization and pruning. The final test results, obtained using our previously designed MWS system and collected dataset of contaminated hazelnut-cocoa spread jars, show a multi-class accuracy for the floating-point model of 96.5%, which corresponds to a binary-equivalent accuracy of 97.3%. This is an improvement of +3.3% against the binary classifier of the previous work. The quantized and pruned model, instead, reached a multi-class accuracy of 94.2%, or a binary accuracy of 95.4%, thus still improving the previous work by +1.4%. Also, we achieved a latency of 26 µs on an AMD/Xilinx Kria K26 FPGA, a result which is ideal for high-throughput food production lines. Furthermore, we expand our latest work with supplementary details and experiments to further validate the proposed ML flow, including a comparative analysis against our prior results. Lastly, we share our datasets publicly on OpenML.**

***Index Terms*—Foreign body detection in food, FPGA acceleration, machine learning, microwave sensing, neural networks**

I. Introduction

AUTOMATION, data-driven insights, and real-time processing are key elements behind the disruptive impact that artificial intelligence (AI) and machine learning (ML) are having in the food industry. The integration of these new technologies into food production lines is rapidly raising industry standards, mitigating the risk of foreign bodies in packaged food [1]. Consequently, consumer health is increasingly safeguarded against foodborne illnesses (e.g., allergies, injuries, choking), whereas food manufacturers experience a reduction in food waste and costly product recall campaigns, enhancing both market reputation and consumer loyalty.

Today, various non-invasive methods exist for the detection of foreign bodies in food, including metal detectors, X-rays, near-infrared, and terahertz imaging [2], [3]. However, these approaches have limitations, such as low-penetration depth, ionizing radiations, strong absorption in water, and inability to detect low-density materials [4]. A promising technology that can overcome these limitations and has demonstrated remarkable detection performance [3], [5]–[8], is Machine-Learning-based Microwave Sensing (MLMWS) [9]. The object to analyze is hit by low-power electromagnetic waves at microwave (MW) frequencies emitted by a set of antennas, which are positioned at some distance from it. Next, the scattered waves are recorded back by the same antennas and the measured scattering parameters are processed by a machine-learning (ML) classifier trained to identify foreign objects [3], [10]. Contaminants are detected thanks to the ability of this method to distinguish two materials in contact based on the difference in their dielectric properties, also known as *dielectric contrast* [3]. Furthermore, MLMWS is particularly suited to very fast food production pipelines in which real-time detection of foreign objects is sufficient and there is no need to run computationally intensive MW image reconstruction algorithms, not to mention the fact that MLMWS is contactless, non-destructive, non-ionizing (and therefore safe for operators), and low-cost [3], [9], [11], [12].

This paper extends our prior works on MLMWS for contaminant detection in the hazelnut-cocoa spread jar case study [3], [9], [13]. Building upon our previous microwave-sensing (MWS) system and dataset [9], we introduce an enhanced ML flow, which is the focus of this paper and is designed to increase the classification accuracy of MLMWS systems. For the details about the design and performance of our MWS system, and the collected dataset, please refer to our previous works [3], [9], [14], [15]. In this work, we use some of the ML techniques employed in [3], [9], like model hyperparameter exploration with *Bayesian Optimization* (BO), model quantization, and model deployment on FPGA with *hls4ml* [16], [17], and bring about a number of new contributions:

1) Differently from all the other MLMWS works, which focus on binary classifiers, we evaluated the detection performance of a multi-class multi-layer perceptron (MLP) classifier, which can determine the presence and

Manuscript received Day Month 2024; revised Day Month 2024; accepted Day Month 2024. This article was recommended by Name Surname. *(Corresponding author: Luca Urbinati.)*

Bernardita Štitić was with the School of Engineering, Pontificia Universidad Católica de Chile, Santiago 7820436, Chile, and with the Department of Electronics and Telecommunications, Politecnico di Torino, Turin 10129, Italy (e-mail: bastitic@uc.cl).

Luca Urbinati and Mario R. Casu are with the Department of Electronics and Telecommunications, Politecnico di Torino, Turin 10129, Italy (e-mail: {luca.urbinati,mario.casu}@polito.it).

Giuseppe Di Guglielmo is with Fermi National Accelerator Laboratory, Batavia, IL 60510, USA (e-mail: gdg@fnal.gov).

Luca Carloni is with the Department of Computer Science, Columbia University, New York, NY 10027, USA (e-mail: luca@cs.columbia.edu).

Digital Object Identifier xx.xxxx/TAFE.2024.xxxxxxx

© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on AgriFood Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAFE.2024.3421238

the type of contaminants. This can help food producers gather statistics on contaminant types not revealed by their quality control systems. We trained the multi-class MLP with features acquired at two different MW frequencies (whereas [9] used only 10 GHz and [3] used eleven frequencies from 9.0 to 11.0 GHz); we augmented our data by adding Additive White Gaussian Noise (AWGN); and we chose a suitable pre-processing scaler (`RobustScaler` instead of `StandardScaler`[1] like in [3], [9]) after a detailed analysis of the feature distributions of our datasets (e.g. Box and Whisker Plots).

2) We employed quantization-aware training (QAT) and pruning (instead [3], [9] just applied post-training quantization) for deployment of the optimal floating-point multi-class model on an AMD/Xilinx Kria K26 FPGA. The compressed model achieved a multi-class accuracy of 94.2% with a latency of 26 µs.
3) We released our datasets [9] on OpenML [18] to encourage future ML research in this field.
4) Specifically, with respect to our latest research [13], we provided additional insights and details on the proposed ML flow, performed new experiments to further validate it, and strengthened the comparison with previous works.

After a summary of the related work in Sec. II, we describe our MWS system and datasets in Sec. III. In Sec. IV, we outline the preliminary experiments that helped us design the flow proposed in Sec. V. Finally, in Sec. VI, we summarize our main findings and comment briefly on potential future work.

II. Related Work

MLMWS is taking hold with several recently published papers. For example, [10], [19], [20] predicted the moisture content in wheat (using MLPs), sweet corn (using MLP, Random Forest (RF), AdaBoost, XGBoost) and peanuts (using MLP, Gradient Boost Regression Trees, XGBoost), respectively; [21] sorted damaged apples with Support Vector Machines (SVMs); [7] distinguished healthy from rancid walnuts using MLPs; [6] analyzed lossy materials mainly made of water by searching for glass and nylon fragments with SVMs; [5] used SVMs for intrusion detection in hazelnut-cocoa cream; [8] trained many ML algorithms (Lasso, AdaBoost, Bagging Tree, Decision Tree, RF, MLPs, ensemble learning, and Graph Neural Network) to detect plastic, paper, wood, glass, aluminum, glue, and cork in several media, both lossy (soda, ice tea, soy sauce) and low-loss (flour, honey). All these works focus on binary classifiers to determine if a contaminant is present or not. In contrast, we developed a multi-class MLP classifier that can also identify the type of contaminant with high precision.

Moreover, our work stands out as one of the few [3], [9], [22] to propose a hardware implementation of the ML algorithm tailored for embedded devices (although [22] does not specifically deal with MLMWS).

III. Microwave Sensing System and Datasets

As explained in Sec. I, this paper proposes an enhanced ML flow for MWS systems. For our experiments, we used the MWS system employed in our previous work [9] and the five datasets gathered at that time[2]. The system, depicted in Fig. 1, comprises a set of six monopole antennas [14] designed to resonate at 10 GHz. These antennas are linked to a 2-port Vector Network Analyzer (VNA) via a custom-made 2×6 electro-mechanical switching matrix [15]. The antennas are mounted on an arch-shaped support above a standard conveyor belt for packaged food. The decisions about the number of antennas, working frequency and mounted support are derived from our previous analysis, which is available in [3]. We do not report the reasons again here, as our focus in this paper is the ML workflow. As the object being examined approaches the arch, a photocell initiates the data acquisition process by triggering a Matlab script running on a laptop. The script configures the switching matrix, triggers a VNA acquisition for each of the 36 combinations of transmitter/receiver antenna pairs (using a single antenna as the transmitter and all the others as receivers), and saves the generated 6×6 scattering matrix (*S-matrix*). The real and imaginary parts [10] of the 15 upper diagonal elements are reshaped into a feature vector with length equal to $N = 30$ that serves as input for the ML classifier. We discarded the monostatic elements at the time of data collection [9], as including these in some of our preliminary image reconstruction experiments led to worse results. Moreover, the majority of the related works [3], [5], [10] adopted the same approach.

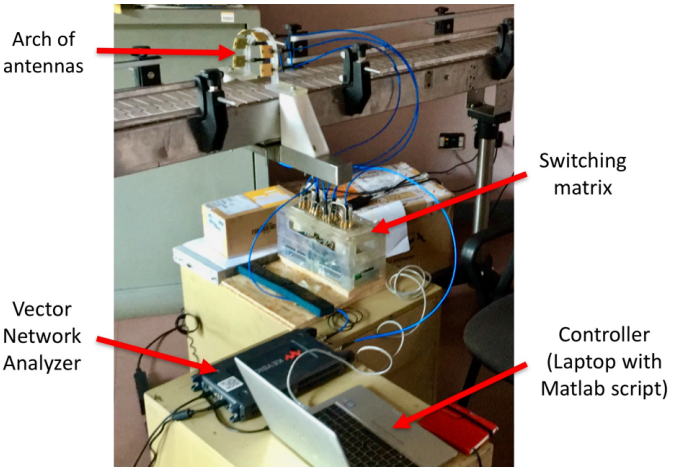


Fig. 1. The MWS system used in this work. Image taken from [9].

Each dataset was obtained by acquiring the S-matrix of 1200 contaminated and 1200 uncontaminated samples of irradiated hazelnut-cocoa spread jars, which we treated as the *positive* and the *negative class* in the binary representation of our problem, respectively. In the multi-class representation, the 1200 contaminated jars were further subdivided into six classes, with 200 samples per class based on each contaminant, while the 1200 uncontaminated ones remained as one class. These contaminants include both high-density and low-density items suggested by a chocolate jar manufacturer (in brackets the max./min. dimensions or the diameter): a cap-shaped plastic (15/9 mm), a glass fragment (13/2 mm), a metal sphere (10

[1]Preprocessing and Normalization: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing. Accessed on: 2024-01-31.

[2]In [9] we gathered five datasets, but we used only one.

© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on AgriFood Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAFE.2024.3421238

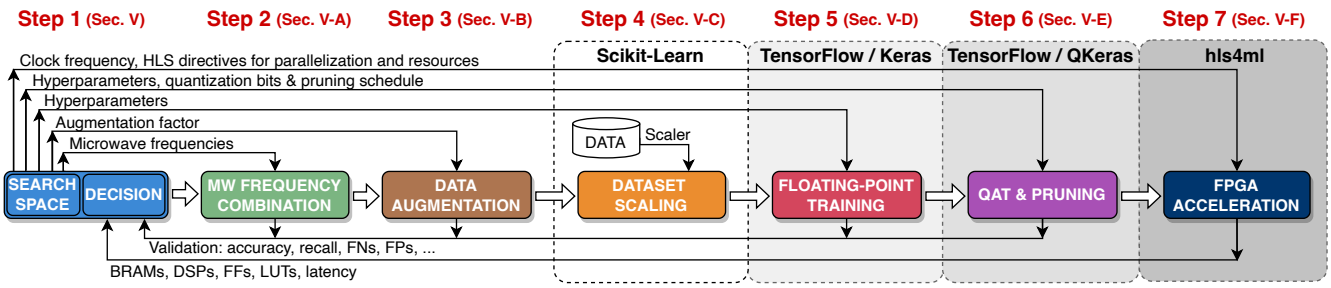


Fig. 2. The proposed enhanced ML flow for food contaminant detection using neural networks trained with data from MWS systems. Modified from [13].

mm), a big plastic sphere (20 mm), a small plastic sphere (3 mm), and a triangular plastic fragment (8/1 mm) (*triangle* in the following). For each contaminant, we varied its position in the jar [9]. Additionally, we acquired all samples at five MW frequencies (9.0, 9.5, 10.0, 10.5, and 11.0 GHz) so that five S-matrices per jar, as well as five distinct datasets, resulted from this process. We published our datasets in the OpenML repository [18] with data IDs 455XX (XX = 36, 37, 38, 39, 40)[3].

IV. Preliminary Machine-Learning Experiments

Here we summarize the preliminary experiments, described in detail in [13], that led to the flow that we present in Sec. V.

We began by analyzing the results obtained with our binary MLP [9], which comprises two hidden layers with 128 and 256 neurons. Once trained on the 10.0 GHz dataset, it provided a good detection accuracy of 94.0% on a balanced Test Set of 480 samples, but it achieved a recall of only 48.8% for the plastic triangle, which corresponds to 21 *false negatives* (FNs) out of 41 samples. This is unacceptable for an industrial scenario, where we ideally need FNs = 0. The reason for this low detection performance, as explained in [9], was due to the position of the triangle during the dataset collection phase, i.e., at the top of the chocolate spread. This created a lower dielectric contrast between plastic and chocolate with respect to that between plastic and air, ultimately making it almost invisible in the back-scattered signal. Therefore, in [13], we explored a multi-class approach for the first time. Indeed, we were convinced that an ML model with multiple outputs could mitigate mispredictions of the triangular shape and, at the same time, help food manufacturers gather statistics on foreign bodies and better identify sources of contamination.

In our preliminary experiments [13], we balanced the 10.0-GHz dataset for the multi-class scenario and developed a multi-class MLP with one hidden layer and 160 neurons. On our balanced Test Set of 280 samples (40 per class), this model reached a multi-class accuracy of 88.6%, or a binary-equivalent accuracy of 98.6%, and a recall of 95.0% (38/40 samples) for the triangle. However, when we added 200 unseen uncontaminated samples to balance our Test Set for the binary scenario, we observed 129 *false positives* (FPs), with 124 free jar samples incorrectly classified as triangle samples, and the multi-class accuracy dropped to 67.1% (–24.3%). From these outcomes, we realized that we had to address three points: 1) the small size of our multi-class sets; 2) overfitting; and 3) our lack of insight into the statistics of our datasets.

We addressed point 1) by adding back all the free jar samples to our 10.0-GHz dataset to maximize data usage. Then, we shuffled and split the entire dataset with stratification, thus obtaining a *New Training Set* (1440 samples) and two equal-sized sets: a *Validation Set* (480 samples) and a *New Test Set* (480 samples). Concerning point 2), we refined the hyperparameter ranges of the multi-class MLP using `StratifiedKFold`[4], which we also used to analyze training and validation curves across folds to detect any split dependent behavior or overfitting. Lastly, we leveraged Scikit-Optimize's `BayesSearchCV`[5], which implements BO with cross-validation, to explore a broad hyperparameter space. We repeated both methods various times as we slightly tuned hyperparameters manually to assess overfitting and validation metrics. In addition, we explored MW frequency combinations (Sec. V-A), data augmentation (Sec. V-B), Max-Norm regularization heuristics [23] and class weights for class imbalance. For point 3), we carefully analyzed our datasets and chose `RobustScaler` as the pre-processing scaler (Sec. V-C).

V. Enhanced Machine-Learning Flow

Based on our previous experiments and the findings presented in Sec. IV, we propose an enhanced ML flow for MWS systems based on a multi-class approach. In particular, we applied this flow to the hazelnut-cocoa spread jar case study as a practical example of food contaminant detection.

The steps of the flow correspond to the *blocks* in Fig. 2. These appear in the order that we recommend executing them. Note that not all of them may be required: for example, an ML designer might not perform data augmentation if the available data is sufficient, or a hardware designer might not need pruning depending on the target FPGA.

The first block, *Search Space and Decision*, is the core of the flow. It contains the user-defined *search space*, which is a collection of all possible combinations of input values for the other blocks. At each flow iteration, the first block: a) outputs a combination of input values from the search space; and b) uses the metrics returned by the other blocks to *decide* on the next

[3]Visit: https://www.openml.org/d/<data_ID>.

[4]StratifiedKFold: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html. Accessed on: 2024-01-31.

[5]BayesSearchCV: https://scikit-optimize.github.io/stable/modules/generated/skopt.BayesSearchCV.html. Accessed on: 2024-01-31.

© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on AgriFood Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAFE.2024.3421238

TABLE I
MULTI-CLASS VALIDATION ACCURACY (%) USING THE FINAL MULTI-CLASS KERAS MODEL OF SEC. V-D TRAINED WITH A SINGLE FREQUENCY (DIAGONAL) OR A PAIR OF FREQUENCIES (IN BOLD, THE COMBINATION SELECTED FOR THE NEXT STEPS OF THE FLOW).

MW freq. (GHz)	9.0	9.5	10.0	**10.5**	11.0
9.0	81.9	89.2	89.2	88.3	89.6
9.5		80.2	87.1	**90.4**	88.8
10.0			80.4	86.3	86.7
10.5				78.1	86.3
11.0					79.0

combination. The execution of these tasks can be either manual or automated (e.g., by performing BO), depending on what aspect of the model needs evaluation (e.g., hyperparameter optimization). In the following we explain the other blocks.

A. Microwave frequency combination

Step two iteratively receives the number of MW frequencies to use from the first block and outputs back the resulting ML metrics of interest (e.g., validation accuracy). This optimization loop helped us explore different combinations of the datasets mentioned in Sec. III to maximize the metrics of interest. The use of multiple frequencies is not new in the literature. Similar findings have been reported in [21], in which the authors observed improvements in their MW imaging system when operating at three frequencies. Consequently, we explored different combinations of the five datasets mentioned in Sec. III and compared the outcomes to those achieved by training on each dataset individually. Table I provides a short summary of the multi-class validation accuracy reached by our final MLP, which we discuss in Sec. V-D, when trained with the scattering parameters of various MW frequencies. By combining features associated with different MW frequencies, we can achieve an accuracy boost which ranges from +4% to +12% in comparison with the same model trained with single-frequency datasets. Based on these results, we selected the 9.5 and 10.5 GHz pair, since it reached the highest validation accuracy. We conjecture that this pair allows the optimization of the frequency response of the antennas. We also trained using combinations of three and four frequencies, but we do not report them in Table I since they resulted in worse performance.

B. Data augmentation

In step three, we augment the Training Set to address model overfitting, enhance generalization and decrease both FNs and FPs. In this case study, the algorithm to generate an artificial sample from an original sample k is presented in the pseudo-code Alg. 1. For each of the $N = 30$ scattering parameters s_i ($i = 1, ..., N$) of k, we computed two normally distributed random variables $v1$ and $v2$ using Gaussian noise with a Signal-to-Noise Ratio $SNR = 60\ dB$ (arbitrary choice). We considered a mean of zero, so as not to introduce bias, and a standard deviation $\sigma = \sqrt{S/SNR}$, where

Algorithm 1 Pseudo-code for generating an artificial sample

Input: Given an original training sample k, with $N = 30$ scattering parameters:
  Compute S;
  Compute σ, with S and $SNR = 60\ dB$;
  for each scattering parameter s_i, with $i = 1...N$ **do**
    Compute $v1$ and $v2$, two Gaussian random variables with zero mean and standard deviation equal to σ;
    Update s_i: $\Re(s_i) \leftarrow \Re(s_i) + v1$; $\Im(s_i) \leftarrow \Im(s_i) + v2$.
  end for
Output: A new artificial training sample is generated from k.

$S = (\sum_{i=1}^{N} \Re(s_i)^2 + \Im(s_i)^2)/N$. Then, we added $v1$ and $v2$ to the real and imaginary parts of s_i, respectively, thus obtaining the new artificial sample.

With this algorithm, we explored gradual increments of the 9.5 & 10.5 GHz Training Set of $M = 1440$ samples. For each original training sample k_j, with $j = 1, ..., M$, we applied Alg. 1 $\alpha - 1$ times to augment the training set α times. In other words, α defines the factor by which we multiply the training set. Moreover, to further decrease the FPs, since the free jars were mostly confused with those containing triangular plastic fragments, we applied Alg. 1 an additional $\alpha_T - 1$ times for each training sample belonging to the triangle class. Therefore, the total augmentation factor for the triangles is $\alpha + \alpha_T$, while for the other six classes it is simply α.

Table II reports the multi-class validation metrics (*T* refers to the triangles and *FJ* to the free jars) obtained after training our final multi-class MLP from Sec. V-D using the 9.5 & 10.5 GHz Training Set, augmented with the (α, α_T) augmentation pair of the first column. As shown in Table II, when gradually increasing α from 1 to 3 with $\alpha_T = 0$ (rows 1–3), FNs remain equal to 9 as FPs decrease from 27 to 20. Then, if we increase α_T to 15 and keep $\alpha = 3$ (row 4), we still obtain FNs = 9 but FPs decrease further from 20 to 16. This is the augmentation pair for which the number of triangular plastic samples equals the number of free jar samples. As soon as we increase α_T to 18 (row 5), we obtain more triangles than free jars and FPs decrease from 16 to 9 at the cost of having FNs increase from 9 to 11. Finally, the last row shows that augmenting all classes by a larger augmentation factor ($\alpha = 4$), without further augmenting the triangles ($\alpha_T = 0$), results in the same recall values and binary FNs/FPs ratio as the ones obtained by using the augmentation pair (3, 15), except for a slight decrease in validation accuracy. This analysis implies that, to improve metrics such as the recall of the triangles and free jars, and find a good trade-off between FNs and FPs, it is necessary to find a suitable relative proportion between the number of triangle and free jar samples in the training set.

Ultimately, we chose the augmentation pair (3, 18), which was the best choice to reduce FPs and balance the number of FNs and FPs. As a result, the New Training Set increased to 6480 samples. Within this set, 2160 samples (33.3%) correspond to free jars, 2520 (38.9%) to triangles and 1800 to the other contaminants, where each one has 360 (5.6%) samples. We define this resulting set as the *Augmented Training Set*.

Next, we validated our choice. As an example, we trained our final model (Sec. V-D) again with the same MW frequency

© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on AgriFood Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAFE.2024.3421238

TABLE II
SELECTION OF MULTI-CLASS METRICS FOR DIFFERENT AUGMENTATION PAIRS, USING THE FINAL MODEL OF SEC. V-D AND THE 9.5 & 10.5 GHZ COMBINATION (IN BOLD, THE PAIR SELECTED FOR THE NEXT STEPS).

Augmentation pair (α, α_T)	**Multi-class Val. acc. (%)**	**Recall T (%)**	**Recall FJ (%)**	**Binary FNs / FPs**
(1, 0)	90.4	77.5	88.8	9 / 27
(2, 0)	91.5	77.5	90.0	9 / 24
(3, 0)	93.8	77.5	91.7	9 / 20
(3, 15)	94.0	77.5	93.3	9 / 16
(3, 18)	**95.0**	**80.0**	**96.3**	**11 / 9**
(4, 0)	93.8	77.5	93.3	9 / 16

TABLE III
MULTI-CLASS RESULTS AS IN TABLE I AND WITH THE AUGMENTATION PAIR (3, 18) OF TABLE II (SELECTED COMBINATION IN BOLD).

MW freq. (GHz)	9.0	9.5	10.0	**10.5**	11.0
9.0	87.3	94.8	93.8	94.2	93.8
9.5		85.8	92.3	**95.0**	92.5
10.0			85.6	91.5	91.7
10.5				84.4	92.1
11.0					86.3

combinations used in Table I. However, this time, we augmented the corresponding training sets with the augmentation pair (3, 18). In Table III, for completeness, we report all the results, not only those for the selected frequency pair. Contrary to the performance reported in Table I, the lowest validation accuracy, which is related to 10.5 GHz in both tables, is now 84.4% rather than 78.1%. On the contrary, the highest accuracy, which was obtained with the 9.5 & 10.5 GHz pair regardless of augmentation, is now 95.0% instead of 90.4%. These results prove the independence between the selection of the MW frequency combination (Sec. V-A) and the choice of the augmentation pair (Sec. V-B). In other words, steps 2 and 3 of our flow can be applied independently and in any order to enhance the performance of the ML classifier.

C. Dataset scaling

In step four, we propose pre-processing the data by using a scaler that addresses its statistics. While this step should be present in every ML flow, it was not present in [9]. We analyzed the histograms of every feature in the five single-frequency training sets (30 features each) and in the combined MW frequency training sets (60 features). We found that most feature distributions did not resemble a Gaussian curve, possibly due to the small size of the datasets. As an example, in Fig. 3(a) we report the histogram of the 21st feature of the 9.5 & 10.5 GHz Augmented Training Set.

In addition, we obtained Box and Whisker Plots to visualize all these feature distributions through quartiles. While we did not find outliers in terms of abnormal values, we did observe data points located past the end of the whiskers. Fig. 3(b) shows Box and Whisker Plots of four representative

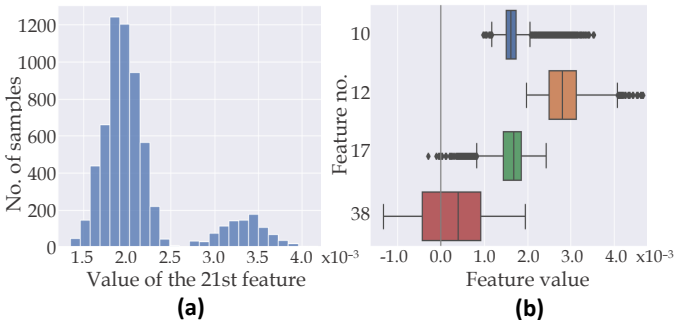


Fig. 3. Examples of feature distributions of the 9.5 & 10.5 GHz Augmented Training Set: (a) a Non-Gaussian distributed feature (the 21st); (b) Box and Whisker Plots of four features (the 10th, 12th, 17th and 38th).

features (10th, 12th, 17th, and 38th) in the 9.5 & 10.5 GHz Augmented Training Set. We observe non-zero median values and some data points located beyond the end of the whiskers. Moreover, we noticed that the ranges of the majority of the features differed by at least one order of magnitude. Finally, concerning the 9.5 & 10.5 GHz combination, we observed that the distribution trends of each feature did not significantly differ from those prior to augmentation.

In light of this analysis, we chose to pre-process the combined 9.5 & 10.5 GHz dataset using Scikit-Learn's `RobustScaler`, which uses statistics that are calculated per feature and are robust to data outliers. For every feature, this scaler subtracts the median of the feature and divides the result by its Interquartile Range, which is the difference between the 75th and 25th percentiles of the feature. Therefore, we first fit the scaler to the Augmented Training Set of 6480 samples to learn the required statistics per feature. Then, we applied the scaling transformation to the Augmented Training Set, as well as to the Validation Set and New Test Set of 480 samples.

D. Floating-point training

The fifth step encompasses some of the steps from Sec. IV, i.e., BO, stratified cross-validation and manual training. We thus obtained our final multi-class MLP with four hidden layers with 300, 151, 195, and 128 neurons, and dropout rates of 0.4, 0.2, 0.1, and 0.05, respectively, with the Scaled Exponential Linear Unit (SELU) activation. We trained our model for 350 epochs, with a batch of 320, the Adam optimizer and a learning rate of 5.5e-5. To address class imbalance, we also employed Scikit-Learn's `compute_class_weight`[6] with the argument `class_weight='balanced'`. In addition, we performed regularization by leveraging the `MaxNorm` Keras class[7] to limit weight norms to 4, as suggested in [23].

On our New Test Set of 480 samples, our classifier achieved a multi-class accuracy of 96.5% and a recall of 85.0% (34/40 samples) for the triangle. Table IV, instead, reports the most relevant binary-equivalent test metrics: our Keras model (row 2) improves the binary MLP of [9] (row 1) by +3.3%.

[6]Compute_class_weight is a function of Scikit-Learn: https://scikit-learn.org. Accessed on: 2024-01-31.

[7]Keras MaxNorm: https://www.tensorflow.org/api_docs/python/tf/keras/constraints/MaxNorm. Accessed on: 2024-01-31.

© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on AgriFood Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAFE.2024.3421238

TABLE IV
TEST METRICS OF THE BINARY VS MULTI-CLASS CLASSIFIERS.

Best MLP classifier	Binary Test. acc (%)	Binary FNs / FPs	Recall T (%)	Recall FJ (%)
Binary ([9])	94.0	21 / 8	48.8	96.7
Multi-class of Sec. V-D (this work)	97.3	6 / 7	85.0	97.1
Multi-class of Sec. V-E (this work)	95.4	7 / 15	82.5	93.8

TABLE V
SELECTION OF VALIDATION METRICS OBTAINED BY TRAINING THE FINAL MULTI-CLASS MLP OF SEC. V-D, AFTER CHANGING SOME DECISIONS MADE IN STEPS 3, 4 AND 5.

Data Aug.	Class weights	Scaler	Max-Norm	Multi-class Val. acc. (%)	Recall T (%)	Recall FJ (%)	Binary FNs / FPs
✓	✓	`Robust`	✓	95.0	80.0	96.3	11 / 9
✗	✓	`Robust`	✓	90.4	77.5	88.8	9 / 27
✓	✗	`Robust`	✓	95.0	75.0	96.7	11 / 8
✓	✓	`Standard`	✓	94.6	77.5	94.6	9 / 13
✓	✓	`Robust`	✗	94.8	77.5	94.2	9 / 14

Finally, we present some sanity-check experiments to validate certain choices that we made to obtain our best floating-point MLP. As a summary, Table V shows some validation metrics derived from training our final model again after modifying specific choices made in steps 3, 4, and 5 of our flow. We note that our floating-point model (row 1) reached the highest results for multi-class validation accuracy and recall of the triangle, together with a good trade-off between FNs and FPs. On the contrary, not augmenting the 9.5 & 10.5 GHz Training Set of 1440 samples (row 2) resulted in the lowest validation accuracy and recall of the free jars, as well as the worst trade-off between FNs and FPs.

As for row 3, training our floating-point model without class weights (i.e., not using `compute_class_weight` as described in Sec. V-D) led to results similar to those in row 1. In particular, validation accuracy is also 95.0%, the recall of the free jars is slightly higher (96.7% instead of 96.3%) and there is a marginal decrease of FPs (8 rather than 9). However, since we aim to maximize the recall of the triangle class, even at the cost of slightly higher FPs, we chose the MLP with class weights because of the higher recall for the triangle (80.0% vs 75.0%). Finally, using `StandardScaler` instead of `RobustScaler` (row 4), or removing Max-Norm regularization as described in Sec. V-D (row 5), decreased the selected metrics in comparison to our best choice (row 1).

E. Quantization-aware training and pruning

In step six, we propose optimizing for hardware resources to improve FPGA deployment (Sec. V-F) with quantization-aware training and pruning in parallel. In particular, we leveraged QKeras for QAT [24] and TensorFlow for pruning. QAT allows for training with reduced bit precision of weights, biases and activations, whereas pruning compresses MLPs by removing irrelevant neurons. Consequently, these methods speed up inference and reduce power consumption [25].

After some exploration [13], for quantization we ended up with: 16 bits for activations (with no integer bits, i.e., all bits represent the fractional part), 8 bits for weights and biases (3 integer and 5 fractional), and $alpha = 1$ as the QKeras scaling factor [24]. Due to quantization, we also removed Max-Norm regularization. Moreover, we changed SELU with `quantized_relu` since the former is not available in QKeras and the latter helped lower FPGA resources (Sec. V-F). In this regard, we also removed the fourth hidden layer. However, we kept SELU over ReLU for our floating-point model as we experienced less overfitting and better detection performance.

For pruning, we selected the `PolynomialDecay` pruning schedule by TensorFlow[8] to prune gradually during QAT. We applied an initial sparsity of 50% at step 2000 and a final sparsity of 75% around step 15120. In this context, steps are obtained by dividing the number of training samples by the batch size, and then multiplying by the number of epochs.

The final quantized and pruned model was trained for 700 epochs, with a batch size of 300 and a learning rate of 5.5e-5. In fact, upon introducing QAT and pruning, we had to fine-tune these hyperparameters with respect to those of Sec. V-D to manage overfitting properly [13]. This model reached a multi-class accuracy of 94.2% on our New Test Set (Sec. IV) and a recall of 82.5% (33/40 samples) for the triangle. However, as expected, its binary-equivalent accuracy decreased to 95.4%, as reported in Table IV (row 3). This represents only a –1.9% decrease compared to the floating-point version of Sec. V-D (row 2) and still surpasses [9] by +1.4% (row 1).

Table VI shows the multi-class confusion matrices for both our floating-point model (top) and its quantized, pruned version (bottom) when predicting on the New Test Set. True and predicted classes are reported along the rows and columns, respectively, in this order: free jars, cap-shaped plastic, glass fragment, metal sphere, big plastic sphere, small plastic sphere, and the triangle. Values on the main diagonal represent correctly predicted samples, while those outside indicate incorrect predictions. Focusing on free jars and triangles (first and last rows/columns, respectively), we observe that the most significant error source in both matrices is the confusion between free jars and triangles. In the confusion matrix obtained using our quantized and pruned model, free jars were affected the most. In particular, the correct predictions, or *true positives* of the free jars, fell from 233 to 225 (–8) with respect to the confusion matrix of our floating-point model (Sec. V-D). Furthermore, FPs increased from 7 to 15 (+8) due to the decrease in the number of correctly predicted free jars, as shown by comparing row 1 of both matrices in Table VI.

Despite this, when comparing to the results achieved by the model of [9], which was not quantized nor pruned, our quantized and pruned model still performs better. In fact, as shown in Table IV, the binary test accuracy improved by 1.4%, a better trade-off between FNs and FPs was achieved, and the recall of the triangle increased by 33.7%. Lastly, regarding

[8]TensorFlow PolynomialDecay: https://www.tensorflow.org/model_optimization/api_docs/python/tfmot/sparsity/keras/PolynomialDecay. Accessed on: 2024-01-31.

© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on AgriFood Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAFE.2024.3421238

TABLE VI
CONFUSION MATRICES OF FLOATING-POINT AND QUANTIZED & PRUNED MODELS ON THE NEW TEST SET. FROM TOP TO BOTTOM AND FROM LEFT TO RIGHT: FREE JARS, CAP-SHAPED PLASTIC, GLASS FRAGMENT, METAL SPHERE, BIG PLASTIC SPHERE, SMALL PLASTIC SPHERE, AND TRIANGLE.

ML classifier	Confusion Matrix
Multi-class Floating-Point model	$\begin{pmatrix} 233 & 0 & 0 & 0 & 0 & 0 & 7 \\ 1 & 39 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 40 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 38 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 40 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 39 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 & 34 \end{pmatrix}$
Multi-class Quantized and Pruned model	$\begin{pmatrix} 225 & 1 & 0 & 0 & 0 & 1 & 13 \\ 1 & 38 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 39 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 39 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 40 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 38 & 0 \\ 6 & 1 & 0 & 0 & 0 & 0 & 33 \end{pmatrix}$

FNs, our models reduce the issue of the plastic triangle mentioned in [9] at the cost of increasing FPs. Nonetheless, FPs are not as critical as FNs for this industrial food scenario.

F. FPGA acceleration

We designed the model described in Sec. V-E for an industrial production line and leveraged FPGA hardware acceleration to reduce latency. Specifically, in our case study, we assumed that the hazelnut-cocoa spread jars moved on a conveyor belt at approximately 30 cm/s, which sets a latency constraint on the ML processing of around 200 ms [9].

We adopted hls4ml, a Python open-source framework [16], [17], [24], to co-design and translate ML algorithms into a hardware implementation while trading-off model accuracy against FPGA resource utilization and inference latency. The hls4ml workflow begins with a floating-point model from a conventional ML framework, such as TensorFlow or PyTorch, or a quantized model from QKeras [26]. Then, it translates the model into C++ code for AMD/Xilinx Vivado HLS [27], which generates a hardware description at the register-transfer level for the traditional synthesis and implementation flow targeting an FPGA as deployment hardware.

Designers can leverage hls4ml to make quick design explorations by configuring the hardware implementation parallelism [28] and, thanks to the integration with QKeras, by also evaluating the impact of low-bit precision on model performance before finalizing a design for FPGA implementation. Whenever tweaking these knobs is insufficient to meet system requirements or to fit the model on the target FPGA, designers may need to revisit previous steps of the flow. This is why we used `quantized_relu` and removed a layer in Sec. V-E.

When translating ML models to hardware, designers must manually specify the accumulator bit precision for each layer of the model to prevent overflow and loss of precision during fixed-point multiply-and-accumulate (MAC) operations. hls4ml offers two solutions: a *dynamic* and a *static approach*.

In the dynamic approach, designers monitor the bit precision of accumulators through simulation on a calibration dataset and adjust the fixed-point format until there is no overflow or loss of model performance. On the other hand, in the static approach hls4ml determines the required fixed-point format without simulation, yet calculates it using the given fixed-point precision of weights, biases, and activations, as well as the number of MAC operations of each layer, which are both known at design-time. The simulation-based approach can be sensitive to the calibration set used for the simulation and results in smaller bitwidths for the accumulators. This may lead to overflow or loss of precision when changing weights in subsequent training campaigns. In contrast, the static approach sizes the bitwidth of accumulators for the worst-case scenario in which activations, weights and biases assume their maximum/minimum representable values. Such edge cases are unlikely to occur in real ML scenarios, therefore the static approach is more conservative and error-free. However, it may require additional FPGA resources for implementing large mathematical operators (e.g., DSPs instead of LUTs). In our exploration, we adopted the described static approach, which preserved in hardware the multi-class test accuracy of 94.2% of the final quantized and pruned model of Sec. V-E.

TABLE VII
COMPARISON OF IMPLEMENTED SOLUTIONS ON FPGA.

Paper	FPGA Name	Tck (ns)	Latency (ms)	BRAMs (%) (#)	DSPs (%) (#)	FFs (%) (#)	LUTs (%) (#)	Power (W)
[9]	Z-7020	10	2.997	43 61	11 25	9 9,576	14 7,448	1.97
This work	K26	5	0.026	2 3	53 656	57 404	94 110,317	3.27

As development system, we chose the AMD/Xilinx Kria K26. This is an ideal solution for edge deployment that combines programmable logic (PL) and an ARM multi-processor in the same system-on-chip (MPSoC). For the final implementation, whose results are reported in Table VII (row 2), we targeted a 5-ns clock period that our model easily reached on the Ultrascale+ FPGA fabric. Our accelerator has a streaming interface and a latency of 809 clock cycles (approximately 4 µs). When integrated with data movers to main memory and a software application to control the accelerator, the overall latency is 26 µs. Finally, the estimated power consumption for the accelerator on the PL is 0.86 W, whereas for the entire chip (including the SoC ARM cores) it is 3.27 W.

VI. CONCLUSION

In this article, we introduced an enhanced ML workflow for MWS systems. In our case study, this approach significantly boosted the accuracy of foreign body detection compared to our previous research. The key innovation contributing to this improvement was the use of a multi-class approach combined with a microwave frequency combination and data augmentation. In the future, we plan to validate our flow across new microwave-sensing problems and automate it completely.

© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on AgriFood Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAFE.2024.3421238

ACKNOWLEDGMENTS

This paper was prepared by Politecnico di Torino, Columbia University, and Fermilab partially using the resources of the Fermi National Accelerator Laboratory (Fermilab), a U.S. Department of Energy, Office of Science, HEP User Facility. Fermilab is managed by Fermi Research Alliance, LLC (FRA), acting under Contract No. DE-AC02-07CH11359.

REFERENCES

[1] K. B. Chhetri, "Applications of artificial intelligence and machine learning in food quality control and safety assessment," *Food Engineering Reviews*, vol. 16, pp. 1–21, 2024.
[2] M. T. Mohd Khairi, S. Ibrahim, M. A. Md Yunus, and M. Faramarzi, "Noninvasive techniques for detection of foreign bodies in food: A review," *Journal of Food Process Engineering*, vol. 41, no. 6, 2018.
[3] M. Ricci *et al.*, "Machine-learning-based microwave sensing: A case study for the food industry," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 3, pp. 503–514, 2021.
[4] J. A. Tobon Vasquez *et al.*, "Noninvasive inline food inspection via microwave imaging technology: An application example in the food industry," *IEEE Antennas and Propagation Magazine*, vol. 62, no. 5, pp. 18–32, 2020.
[5] A. Darwish *et al.*, "Physical contamination detection in food industry using microwave and machine learning," *Electronics*, vol. 11, no. 19, 2022.
[6] ——, "In-line microwave nondestructive evaluation of packaged food products via the support vector machine algorithm," in *Proc. Int. Symp. on Antennas and Propagation and USNC-URSI Radio Science Meeting*. Portland, Oregon, USA: IEEE, 2023, pp. 343–344.
[7] E. Kızılay, C. Aydinalp, and M. N. Akinci, "Neural network-based classification for walnut state using microwave scattering parameters," *TechRxiv*, 2023. [Online]. Available: http://dx.doi.org/10.36227/techrxiv.23904462.v1
[8] M. Musumeci *et al.*, "Development of a deep-learning pipeline to detect and locate contaminants of industrial products via non-invasive microwave signals," in *Proc. Conf. on AgriFood Electronics (CAFE)*. Turin, Italy: IEEE, 2023, pp. 45–49.
[9] L. Urbinati *et al.*, "A machine-learning based microwave sensing approach to food contaminant detection," in *Proc. Int. Symp. on Circuits and Systems (ISCAS)*. Seville, Spain: IEEE, 2020, pp. 1–5.
[10] P. Bartley, S. Nelson, R. McClendon, and S. Trabelsi, "Determining moisture content of wheat with an artificial neural network from microwave transmission measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 47, no. 1, pp. 123–126, 1998.
[11] "Food radar," https://www.foodradar.com/, Accessed on: 2024-01-31.
[12] "Wavision," https://www.wavision.it/, Accessed on: 2024-01-31.
[13] B. Štitić *et al.*, "Enhanced machine-learning flow for microwave-sensing systems to detect contaminants in food," in *Proc. Conf. on AgriFood Electronics (CAFE)*. Turin, Italy: IEEE, 2023, pp. 40–44.
[14] M. Ricci, J. A. T. Vasquez, R. Scapaticci, L. Crocco, and F. Vipiana, "Multi-antenna system for in-line food imaging at microwave frequencies," *IEEE Transactions on Antennas and Propagation*, vol. 70, no. 8, pp. 7094–7105, 2022.
[15] J. A. Tobon Vasquez *et al.*, "Design and experimental assessment of a 2D microwave imaging system for brain stroke monitoring," *International Journal of Antennas and Propagation*, vol. 2019, pp. 1–12, 2019.
[16] FastML Team, "fastmachinelearning/hls4ml," https://github.com/fastmachinelearning/hls4ml, Accessed on: 2024-01-31.
[17] J. Duarte *et al.*, "Fast inference of deep neural networks in FPGAs for particle physics," *Journal of Instrumentation*, vol. 13, no. 7, p. P07027, 2018.
[18] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2014.
[19] J. Zhang *et al.*, "Development of multifrequency-swept microwave sensing system for moisture measurement of sweet corn with deep neural network," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 9, pp. 6446–6454, 2020.
[20] F. Ma *et al.*, "Determining peanut moisture content by scattering coefficient," *Journal of Food Engineering*, vol. 344, 2023.
[21] F. Zidane *et al.*, "Damaged apple sorting with mmwave imaging and nonlinear support vector machine," *IEEE Transactions on Antennas and Propagation*, vol. 68, no. 12, pp. 8062–8071, 2020.
[22] D. Kolosov *et al.*, "Microbiological quality estimation of meat using deep CNNs on embedded hardware systems," *Sensors*, vol. 23, no. 9, 2023.
[23] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.
[24] C. N. Coelho *et al.*, "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors," *Nature Machine Intelligence*, vol. 3, no. 8, pp. 675–686, 2021.
[25] A. Gholami *et al.*, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision: Improve the Efficiency of Artificial Intelligence*, 1st ed. New York, NY, USA: Chapman and Hall/CRC, 2022, ch. 1.2.12, pp. 14–17.
[26] Google, "QKeras notebooks," https://github.com/google/qkeras/tree/master/notebook, Accessed on: 2024-01-31.
[27] AMD, "Vivado HLS," https://www.xilinx.com/products/design-tools/vivado/high-level-design.html, Accessed on: 2024-01-31.
[28] F. Fahim *et al.*, "hls4ml: An Open-Source Codesign Workflow to Empower Scientific Low-Power Machine Learning Devices," in *Proc. tinyML Research Symposium*, 2021.

VII. BIOGRAPHY SECTION

Bernardita Štitić received the B.S.E. degree (highest honors) from Pontificia Universidad Católica de Chile (PUC) in 2019, and two M.Sc. degrees (both summa cum laude) in electronic engineering from Politecnico di Torino, with a focus on embedded systems, and Politecnico di Milano in 2020. She also received her Professional Title in electrical engineering (highest honors) from PUC in 2022. She is currently a software engineer at one of the five largest Big Tech companies, in Redmond, WA, USA. Her research interests lie in artificial intelligence and machine learning for practical applications.

Luca Urbinati (GSM'21) received the B.Sc. degree in Electronics and Telecommunications from the Università di Bologna, Italy, in 2017, and the M.S. degree in Electronic Engineering from the Politecnico di Torino, Italy, in 2019, where he is currently pursuing the Ph.D. His research focuses on integrated architectures for AI and machine learning, targeting embedded edge devices through HW-SW co-design. He has collaborated across disciplines, including food safety, embedded systems, and high-level synthesis.

Giuseppe Di Guglielmo received the Laurea degree (summa cum laude) and the Ph.D. degree in computer science from the Università di Verona, Verona, Italy, in 2005 and 2009, respectively. He is currently a senior ASIC engineer at Fermi National Accelerator Laboratory. He actively contributes to the open-source SoC design and ML acceleration communities.

Luca Carloni (Fellow, IEEE) received the Laurea (summa cum laude) degree in electrical engineering from the Università di Bologna, Bologna, Italy, in 1995, and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 1997 and 2004, respectively. He is currently a Professor and Chair of Computer Science with Columbia University, New York City, NY, USA. His research interests include heterogeneous computing, system-on-chip platforms, embedded systems, and open-source hardware. He authored or co-authored over 180 publications. He is a Senior Member of the Association for Computing Machinery. He was selected as an Alfred P. Sloan Research Fellow in 2008. He was a recipient of the Demetri Angelakos Memorial Achievement Award in 2002, the NSF Faculty Early Career Development (CAREER) Award in 2006, the ONR Young Investigator Award in 2010, and the IEEE CEDA Early Career Award in 2012. In 2013, he was the General Chair of Embedded Systems Week, the premier event covering all aspects of embedded systems.

Mario R. Casu (Senior Member, IEEE) received the Ph.D. degree in electronics and communications engineering from the Politecnico di Torino, Torino, Italy, in 2001, where he currently is Associate Professor. His research interests are systems-on-chip with specialized accelerators, system-level design and design methodology for FPGAs and ASICs, and embedded machine learning. He is also interested in the design of circuits, systems, and platforms for industrial applications, such as biomedical, automotive, and food. His past work focused mostly on latency-insensitive design of systems-on-chip (SoC) and networks-on-chip. He regularly serves in the Technical Program Committee for international conferences, such as DAC, ICCAD, and DATE.

© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.